UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

TECHNOLOGY PROPERTIES LIMITED and PATRIOT SCIENTIFIC CORPORATION, Plaintiffs, v. MATSUSHITA ELECTRIC INDUSTRIAL CO; LTD; PANASONIC CORPORATION OF NORTH AMERICA; JVC AMERICAS CORP.; NEC CORPORATION; NEC ELECTRONICS AMERICA, INC.; NEC AMERICA, INC.; NEC DISPLAY SOLUTIONS OF AMERICA, INC.; NEC SOLUTIONS AMERICA, INC.; NEC UNIFIED SOLUTIONS, INC.; TOSHIBA CORPORATION; TOSHIBA AMERICA, INC.; TOSHIBA AMERICA ELECTRONIC COMPONENTS, INC.; TOSHIBA AMERICA INFORMATION SYSTEMS, INC.; and TOSHIBA AMERICA CONSUMER PRODUCTS, LLC, Defendants.	Civil Action No. 2-05CV-494 (TJW) JURY TRIAL DEMANDED

DEFENDANTS' BRIEF REGARDING CONSTRUCTION OF DISPUTED CLAIM TERMS OF THE 584 PATENT

Table of Contents

Table of Authorities

Cases

I. OVERVIEW OF THE 584 PATENT.

The inventors of the '584 patent faced a problem recognized by computer designers since the earliest days of computing: processors could frequently process computer instructions faster than those instructions could be loaded from memory into the processor. As the '584 patent notes: "The bottleneck in most computer systems is the memory bus." (4:3-4).[1] Claim 29 of the '584 patent claims a method to address this bottleneck.

As this Court recently recognized, "the interpretation to be given a term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim." *Biax Corp. v. Intel Corp.*, 2007 WL 677132 at *3 (E.D.Tx. 2007). To understand how to construe what the '584 inventors might have invented at the time of the 1989 filing, it is important to understand what they did not invent:

- They did not invent the concept of fetching multiple instructions from memory to the processor in a single fetch – computer pioneer Seymour Cray used this concept in the 1960's and others used it in numerous subsequent systems.
- They did not invent Reduced Instruction Set Computers ("RISC"), which are generally attributed to researchers at Stanford University and the University of California-Berkeley starting in 1980. The '584 patent cites "RISC microprocessors ... exemplified by the Sun SPARC." (1:31-32). The Berkeley and Stanford pioneers created "SPARC" and "MIPS" RISC computers, respectively.
- They did not invent the combination of fetching multiple instructions plus RISC. Both SPARC and MIPS machines had this combination before the '584 patent.

Rather, the '584 inventors addressed the memory bottleneck by adapting the well-known idea of fetching multiple instructions during a single memory fetch with another concept. Namely, if some multiply-fetched instructions could appropriately be grouped in memory, there would be "opportunities to optimize groups of instructions" that had been fetched together. 22:62-66.

[1] All citations to patents are from the '584 patent and in the form Column: Line(s).

Thus, the '584 inventors optimized program execution by arranging certain instructions to load with related instructions into the Instruction Register of the processor as a group. In essence, the grouped instructions in the Instruction Register could function almost as a "mini-program within a program." See, for example, the "microloop" example in column 24 at lines 22-36 in which several *related* instructions execute rapidly in a loop because those instructions were all loaded as an "instruction group" into the Instruction Register. Because the Instruction Register is entirely contained on the processor, these mini-programs of instructions within the Instruction Register avoid the memory bottleneck because new instructions need not be fetched from memory during the execution.

The inventors came to call these "opportunities to optimize groups of instructions" under the coined term "instruction groups" over the course of lengthy proceedings with the PTO. In explaining the '584 patent claims, the specification and prosecution history extensively defined "instruction groups" to include:

- "Right justified operands": the specification and prosecution history repeatedly called this the "**magic**" that "**always**" and "**must**" occur;
- "Groupedness": this reflects the concept that the "optimized" instructions in an instruction group are loaded into the Instruction Register with related instructions;
- Distinction from single instructions: As Plaintiffs state, an "important feature of the '584 patent [is] that the instruction register holds an 'instruction group' rather than the traditional single instruction." Plaintiffs' Br. at 39.

These necessary characteristics separate an "instruction group" from the standard "group of instructions" present in every computer program ever written.

Accordingly, the meaning of the term "instruction group" is crucial to understanding what is the novel invention of the '584 patent. As such, it is the key disputed term between the parties with respect to claim 29, and is the focus of Defendants' brief.

The parties agree that claim 29 requires reference to the intrinsic record. Consistent with that intrinsic record, Defendants' proposed claim construction for "instruction groups" is:

> sets of from 1 to a maximum number of sequential instructions, in which the execution of the instructions depends on each set being provided to the instruction register as a unit and in which any operand that is present must be right justified and which cannot encompass a single 32-bit **traditional conventional** instruction."[2]

In contrast, Plaintiffs ignore the intrinsic evidence with respect to the critical term "instruction group" and thereby avoid strong language in the specification and prosecution history that recites the "magic" of the invention which "must" "always" be present. Plaintiffs rely on extrinsic evidence – the conclusory declaration of their retained expert Despain – to attempt to contradict the clear and unambiguous intrinsic evidence. In so doing, Plaintiffs propose a claim construction of "instruction group" to mean "sets of from 1 to a maximum number of sequential instructions, each set being provided to the instruction register as a unit and having a boundary." Nothing in Plaintiffs' proposed construction distinguishes "instruction group" from the ordinary "group of instructions" found in every single computer program – as was specifically noted by the PTO examiners.

The balance of the arguments is simple: a single conclusory section of paragraphs from Plaintiffs' expert mimicking Plaintiffs' legal argument versus extensive passages from the specification and prosecution history. Such a balance tips heavily in the Defendants' favor and the Court should construe the coined term "instruction group" as Defendants propose.

II. DISPUTED TERMS FROM THE '584 PATENT.

For the Court's convenience, Exhibit AA contains a table listing the disputed claim terms, both parties' proposed construction and Defendants' proposed modified constructions based on Plaintiffs' statements in their opening brief.

A. "Instruction Group."

The parties agree that "instruction group" is not a term of art and must be construed as the patentees defined that term in the intrinsic record. Plaintiffs' definition fails to recognize the

[2] As noted below, the "**traditional conventional**" term is taken directly from Plaintiffs' brief and replaces the term "**RISC**" in Defendants' proposed construction.

extensive exchange in which the PTO repeatedly rejected the '584 claims until the patentees gave "instruction group" a very specific definition. Defendants' construction gives meaning to three concepts found extensively in the intrinsic record:

- "Right justified operands," which were repeatedly called the "**magic**" of the invention that "**always**" and "**must**" occur. It would be unusual to exclude from the definition of a coined term such as "instruction group" a concept that the patentees repeatedly described in such strong terms. Notably, Plaintiffs omit all discussion of the extensive *intrinsic* record on "right justified operands" in favor of a single sentence citing to the *extrinsic* conclusory statement of an expert.
- "Groupedness," which was another term coined by the patentees and embodies the idea that "instruction groups" are not arbitrary groups of sequential instructions. Rather, "instruction groups" are optimized sets of instructions that depend on the specific members of the "instruction group" being loaded together.
- Recognizing that the term "instruction *group*" does not encompass a "traditional single instruction" or a "single instruction as is conventional." Notably, Plaintiffs' brief (at 39) expressly distinguished instruction "groups" from a "traditional single instruction" and a "single instruction as is conventional." Defendants adopt Plaintiffs' distinguishing construction in this regard.

Defendants respectfully request the Court adopt their definition of "instruction group" as the only one giving any meaning to the intrinsic record:

> sets of from 1 to a maximum number of sequential instructions, in which the execution of the instructions depends on each set being provided to the instruction register as a unit and in which any operand that is present must be right justified and which cannot encompass a single 32-bit RISC [or traditional conventional] instruction.

The highlighted passages represent the necessary constructions to give life to the three concepts above defined by the patentees as being necessary to be an "instruction group."

1. Right Justified Operand.

The *intrinsic* record directly and repeatedly confirms that the patentees defined "instruction group" to include that all operands[3] in an instruction group be right-justified and such a definition was necessary to obtain the asserted claim. Plaintiffs argue this issue in a single sentence (p. 40) that cites solely to *extrinsic* evidence in the form of an *ipse dixit* expert statement. Plaintiffs' attempt to avoid substantive discussion highlights the strength of the Defendants' position.[4]

The '584 patentees' own words state the requirement of right-justified operands in stronger words than any brief could use. Starting with the specification:

> This **magic** is possible because operands **must** be **right justified** in the instruction register. This means that the least significant bit of the operand is **always** located in the least significant [right-most] bit of the instruction register.

16:14-18.[5] This unequivocal statement was not an isolated passage. The patentees emphasized this "magic" statement twice in efforts to overcome prior art rejections. See Ex. DD at 13 [6/17/97 Amendment at p. 13]; Ex. GG at 7 [2/5/1998 Amendment].

The patentees' use of such clear-cut words like "magic," "must," and "always" in defining the meaning of (the non-standard term) "instruction group" was unambiguous. Indeed, this language is strikingly similar, but much stronger, than the language in *SciMed Life Sys., Inc. v. Advanced Cardiovascular Sys., Inc.*, 242 F.3d 1337, 1343 (Fed. Cir. 2001)("structure defined above is the basic . . . structure for all embodiments of the present invention contemplated and disclosed herein") which compelled adoption. Just as in *SciMed*, "[i]t is difficult to imagine how the patents could have been clearer in making the point." *Id*. at 1343. The patentees should not be permitted to repeatedly tell the PTO that a certain aspect is "magic" which "must" "always" occur and then evade that language during litigation.

---

[3] In very generalized computer terms, operands are data elements on which the processor operates.

[4] For the Court's convenience, Ex. BB has examples representing the "right justification" "magic" that "must" "always" be present in an instruction group contrasted with an ordinary group of instructions.

[5] Unless otherwise noted, all emphases added.

Even beyond the "magic" of the '584 patent, the patentees repeatedly expressed the necessity of "right justified operands" in other unambiguous terms in virtually every single substantive interaction with the PTO. For example, the patentees stated:

> The processor in the instant case solves this problem by placing the **operands always in the same place within an instruction group** so they are in a **fixed place**… by essentially placing operands and flowing instructions around them… the implementation simplicity of **fixed location operands** is also achieved.

Ex. DD at 10 [6/17/97 Amendment]. The "always in the same place," "fixed place," and "fixed location" for the operands are the right-justified location in the instruction group as confirmed later in the same amendment:

> The **operand** for the instruction includes the **bits from the op-code to the last bit in the current instruction group.**

Ex. DD at 13 [6/17/97 Amendment]. In order for the "operand" to be in the "last bit in the current instruction group," the operand is necessarily right-justified.

> Branches do not use a given number of operand offset bits, but **use the bits remaining in the instruction group**.

Ex. DD at 8 [6/17/97 Amendment]. The "remaining bits" are those bits that extend to the "right" side of the instruction group – thus making the "remaining bits" right-justified.

In the third amendment, the patentees left no doubt that to overcome the prior art "operands are right justified" was necessary:

> corresponding language in claims 91 and 97 [prosecution claim 97 corresponds currently asserted claim 29] has been rewritten, both to make the language more definite and to define the invention better over the prior art, … As pointed out in the interview, by establishing such a predefined location for the operand and instructions, processor construction is simplified…. Operands are right justified within the instruction register so they are already properly aligned for use.

Ex. FF at 8-9 [11/26/97 Amendment].

Finally, in the fourth amendment, the patentees reprised the "magic" argument from the specification and used additional strong language to define the "instruction group:"

> "This **magic** is possible because operands **must** be right justified in the instruction register. This means that the least significant bit of the operand is **always** located in the least significant bit of the instruction register." Thus the operand is loaded from the **end** boundary of the current instruction group.

See Ex. GG at 7 [2/5/98 Amendment]. There is no dispute that the "end" boundary is the "right" boundary in the context of the '584 patent.

To oppose this overwhelming intrinsic evidence, Plaintiffs cite (at 40) only a single conclusory paragraph in the extrinsic declaration by their expert, Dr. Despain. Even that conclusory paragraph is a very weak statement:

> 116. I believe that right justified operands are not required by the specification or the file history … But many of the instructions will provide the claimed behavior even if operands are not always right justified as Defendants would require.

Plaintiffs' reliance on such a statement is faulty for a number of legal and factual reasons. Legally, *Phillips* forecloses any reliance on such an unreasoned conclusory statement:

> However, conclusory, unsupported assertions by experts as to the definition of a claim term are not useful to a court. Similarly, a court should discount any expert testimony "that is clearly at odds with the claim construction mandated by the claims themselves, the written description, and the prosecution history, in other words, with the written record of the patent."

Phillips v. AWH Corp., 415 F.3d 1303, 1318 (Fed. Cir. 2005)(citation omitted). Plaintiffs' expert provides no analysis of the extensive intrinsic record "that is clearly at odds" with his unsupported assertions. Additionally, it is irrelevant whether instructions will provide the same behavior. The question for claim construction is not whether some same behavior can be provided, but rather, whether statements were made in the intrinsic record defining the non-standard term "instruction group."

Factually, Despain's statement that many instructions "will provide the same behavior" is untrue. Notably, Despain fails to identify any of these alleged "same behavior" instructions. Despain's inability to cite *any* support is highly indicative that there is no written description supporting such a construction and that his *ipse dixit* claim construction should therefore be ignored.

The Court should reject Plaintiffs' attempt to re-write the contemporaneous intrinsic record with the unreasoned statement of an expert that contradicts the written record. "Expert testimony in conflict with the intrinsic evidence, however, should have been accorded no

weight." *Desa IP, LLP v. EML Techs., Ltd.*, 2007 U.S. App. LEXIS 256, *12 (Fed. Cir. 2007). "*Phillips* rejected any claim construction approach that sacrificed the intrinsic record in favor of extrinsic evidence, such as dictionary definitions or expert testimony." *Biax Corp. v. Intel Corp.*, 2007 WL 677132 at *3 (E.D.Tx. 2007).

The Court need only balance two competing proposals: a single conclusory section of paragraphs from Plaintiffs' expert mimicking Plaintiffs' legal argument versus extensive passages from the specification and prosecution history. Such a balance tips heavily in the Defendants' favor and the Court should construe the coined term "instruction group" to include that "any operand that is present must be right justified."

2. What The Patentees Coined As "Groupedness."

Another characteristic of the coined term "instruction group" is that the particular bytes in instruction groups must necessarily be "grouped" together for successful execution of the instructions. Plaintiffs portray this concept as a "boundary" limitation, but such a definition does not accurately describe the "unique characteristic" that the patentees called "groupedness" to the PTO. Under that concept, an "instruction group" cannot just be a "group of instructions" because such a definition has no distinction over every computer program ever written. Rather "groupedness" indicates some relationship between the instructions within the group and is embodied by Defendants' proposed language of "in which the execution of the instructions depends on each set being provided to the instruction register as a unit."

As an initial matter, the PTO already rejected Plaintiffs' proposal to "simply requir[e] that each group has a boundary:"

> [I]t is well know[n] that instructions in all computer programs, including [the cited prior art], are arranged in sequential order. In other words, all instructions within a computer program are located relative to each other or to each other groups [sic] if they are logically grouped… An instruction being located relative to an instruction group is not patentable over the applied art.

Ex. EE at 3-4 [8/22/97 Office Action]. Thus, "instruction group" cannot mean just an ordinary group of sequential instructions clumped together with some arbitrary boundary because that definition would describe "all computer programs." In other words, simply having a "boundary"

is not enough: that boundary must have some significance to the processing of (at least some of the) instruction groups.

Responding to the PTO's rejection of defining "instruction group" as just groups of sequential instructions, the patentees identified specific "instructions" that represented the concept of "groupedness" in the third and fourth amendments.

> One of the unique characteristics of the claimed processor and processing method is the locating of operands or instructions by their position within the current group of instructions, or a characteristic we call "groupedness." This characteristic is represented in the disclosure by example in various instructions that are the subject of the dependent claims.

See Ex. GG at 6 [2/5/98 Amendment]; Ex. FF at 8-11 [11/26/97 Amendment]. These "groupedness" instructions were: Skip, Microloop, Jump-type (branch), Fetch-via-PC and Load-Short-Literal. *Id*. Each of these instructions relies upon the particular set of bytes for a "group" to be necessarily loaded together in the instruction register together:

- Skip: "SKIP instructions in the microprocessor skip any remaining instructions in a 4-byte instruction group." 23:7-9. As described in the patent, a "Skip" is used when a certain test is met during the execution of the patent. *E.g.*, 23:33-35 ("If the TOP item of the Parameter stack has a most significant bit equal to '0', skip any remaining instructions in the 4-byte instruction group."). Thus, the act of "Skipping" establishes a relationship between the "test" and the skipped instructions. The Skip instruction must be loaded into the instruction register as a group with those bytes that the program execution knows can be safely skipped if the test conditions are met. "Skip" cannot just skip random bytes.
- Microloop: "Microloops are a unique feature of the microprocessor architecture which allows controlled looping within a 4-byte instruction group. A microloop instruction tests the LOOP COUNTER for "0" and may perform an additional test. If the LOOP COUNTER is not "0" and the test is met, instruction execution continues with the first instruction in the 4-byte instruction group, and the LOOP COUNTER is decremented. …

If the LOOP COUNTER is "0" or the test is not met, instruction execution continues with the next instruction." 24:1-12. To correctly "loop" over the desired instructions, those instructions must necessarily be loaded into the instruction register together as a group. As with the Skip command, looping through random instructions makes no sense.

- Jump-type (branch): These instructions direct the processor to "jump" to another point in memory with the help of an "operand" to continue execution at the new point. The "jump" instruction is necessarily loaded into the instruction group with the operands that help define the address to "where" the processor jumps to continue execution.
- Fetch-Via-PC: "Fetch the 32-bit memory content pointed to by the Program Counter and push it onto the Parameter Stack." 26:65-27:1 In defining the meaning of "groupedness" for Fetch-Via-PC, the patentees clarified that the referenced "instruction group" was the target 32-bits loaded by this instruction (*e.g.*, Plaintiffs' Br. at 43 making the exact same argument):

  > the operand, the 32-bit memory content fetched, is …, the next instruction group, and fills an entire instruction group.

  In other words, the pertinent "instruction group" is the "next instruction group" and it "fills an entire instruction group." Thus, the "next instruction group" must necessarily have the correct 4 bytes (32-bits) that the program needs to "push [] onto the Parameter Stack" – it *cannot* just be some arbitrary 32-bits having some arbitrary "boundary" as implied by Plaintiffs' construction. It is a specific 32-bits that must be loaded together.
- Load-short-literal: "Push the 8-bit value found in byte 4 of the current 4-byte instruction group onto the Parameter Stack." 29:22-24. As this definition states, execution of the Load-short-literal command necessarily relies upon the "8-bit value" being loaded into the instruction register with the "current 4-byte instruction group."

Importantly, Plaintiffs do not, cannot, and will not be able to, deny these key exemplary instructions – which were repeatedly cited to the PTO as representing the concept of

"groupedness" – rely upon the correct "group" of bytes being loaded into the instruction register as an "instruction group" for proper execution.[6]

The key flaw in Plaintiffs' position is revealed in Plaintiffs' argument (at 40) that "execution of logic and math instructions… is not affected by group boundaries." While this statement it true, it is irrelevant to the "instruction group" definition because none of these logic and math instructions are ever referred to by the specification or prosecution history as having the concept of "groupedness." These "logic and math instructions" are not the instructions constituting the instruction groups "represented [to the PTO] in the disclosure by example in various instructions." Ex. GG at 6 [2/5/98 Amendment].

Thus, Plaintiffs contradict the intrinsic record in stating (at 40) that requiring "every group" to have this characteristic "is too strong" to be considered an "instruction group." This erroneous view misstates the fundamental difference – required by the PTO – in the meaning of "instruction groups" versus "groups of instructions." It is true that not every "group of instructions" need have the characteristic of being necessarily loaded together in the instruction register – the exact reason that the patentees did not cite the "logic and math instructions" for the concept of "groupedness." However, the only instructions cited and argued under the rubric of "groupedness" are those in which the execution of the instructions depends on each set of the "instruction group" being loaded as a unit.

As noted by the PTO examiner, a "group of instructions" is not an "instruction group" as used by the asserted claim. The specification noted this dichotomy by noting an "advantage" as being "[o]pportunities to optimize groups of instructions." 22:60-66. It is these "optimized" groups of instructions that are "instruction groups."

Plaintiffs' construction, merely requiring arbitrary "boundaries," makes no sense. What does "having a boundary" mean? Everything has "boundaries" and every group of randomly-

[6]Plaintiffs' Br. at 42-43 cites how the same seven instructions "consistently" operated in a certain manner and that the consistent manner should be considered a characteristic of the claim. Defendants merely request that the same analytical approach be recognized with respect to the need for particular related bytes to be loaded together.

selected bytes in a computer program has "boundaries." Merely saying that an instruction group has "boundaries" gives no life to the coined term of "instruction group," the concept of "groupedness," or the specific instructions cited to the PTO for that concept.

3. Distinguishing "*Groups*" From *Single* Instructions.

The coined term "instruction group" cannot include single instructions as found in the prior art – such a construction gives the term "group" no meaning. Indeed, such a construction entirely writes the term "group" out of the claim. Plaintiffs' brief recognized this exact concept when discussing "Instruction Register."

> Plaintiffs' construction also incorporates the important feature of the '584 patent that the instruction register holds an "instruction group" rather than the traditional single instruction. This construction is based on the language of claim 29 itself, which recited "providing **instruction groups** to said instruction register," implying that the IR receives and holds a group rather than a single instruction as is conventional.

Plaintiffs' Br. at 39 (underlined emphasis added). This passage states Defendants' point: "an instruction group" is different from the "traditional single instruction" and the Instruction Register "holds a group rather than a single instruction as is conventional." Defendants embody this same principle with the language "and which cannot encompass a single 32-bit RISC instruction" in Defendants' original definition of instruction group.

Defendants' adopt Plaintiffs' characterization of this concept. The concept can be expressed in one of two manners:

(1) When the "Instruction Register" holds a "traditional single instruction" or a "single instruction as is conventional," the Instruction Register is not holding an "instruction group."

Or

(2) The definition of "instruction group" excludes the "traditional single instruction" and the "single instruction as is conventional" by including Defendants' proposed language that states "and which cannot encompass a single 32-bit ["RISC" or "traditional conventional"] instruction."

Either expression of this concept is acceptable. Defendants characterized this concept as exclusion of a "single 32-bit RISC instruction" but Plaintiffs' characterization as excluding a "traditional single instruction" or a "single instruction as is conventional" is acceptable.

Finally, the intrinsic record supports distinguishing "instruction group" from single 32-bit instructions (regardless of characterization as RISC). The specification states:

> The microprocessor 50 fetches instructions in 32-bit chunks called 4-byte instruction groups. These four bytes may contain four 8-bit instructions or some mix of 8-bit and 16 or 24-bit instructions.

23:4-7. This statement defines that "group" means other than a single 32-bit instruction.

Plaintiffs' statement that "[t]he '584 patent specifically includes some 32-bit instructions" is true, but irrelevant to the definition of "instruction group." The specification never identifies these 32-bit *single* instructions as "instruction *groups*" (which would be a rather strained definition of the word "group"). Rather, as is consistent with the common ordinary meaning of group, the specification defined "instruction group" as being "8-bit and 16 or 24-bit instructions." 23:4-7.

The specification statement that "[a]n instruction group may contain from one to four instructions" (19:17-18) is consistent with Defendants' construction. These "one to four instructions" are "four 8-bit instructions or some mix of 8-bit and 16 or 24-bit instructions" (23:4-7) not the 32-bit "traditional single instruction" "as is conventional" (Plaintiffs' Br. at 39). Indeed, the specification explicitly tied these "shortened" instructions to the optimized groups of instructions ("instruction groups"):

> [m]ost of the work in the microprocessor 50 is done by the 8-bit instructions… the availability of 8-bit instructions also allows another architectural innovation, the fetching of four instructions in a single 32-bit memory cycle. The advantages of fetching multiple instructions are: …. Opportunities to optimize groups of instructions.

22:42-66 (distinguishing 32-bit instructions). The idea that an instruction group "may contain one to four instructions" corresponds with the unambiguous language that those instructions are either 8-bit, 16-bit, or 24-bit instructions (perhaps with right-justified operands to make them an "instruction group") but not a 32-bit "traditional single instruction" "as is conventional."

B. "Operands."

The Defendants' proposed construction recognizes for the definition of "operand" in the context of the '584 intrinsic record that:

> an input to an operation specified by an instruction that is encoded as part of the instruction and the size of the [operand] *can* vary depending on the value of the operand.

Plaintiffs mischaracterize this underlined option as requiring "only variable-width operands" (at 41). In fact, however, Defendants' proposed construction merely provides – consistent with the intrinsic record – that the operands have the capability to be variable width in some instances.

The '584 patent uses variable width operands throughout to minimize the number of bytes storing the operand. For example, if the value "1" is used, the '584 teaches that the "1" is represented by a single byte (*e.g.*, 00000001) rather than multiple bytes (*e.g.*, 00000000 00000001). Such an improvement decreases the overall code size and can be more efficient.

The intrinsic record clarifies that the ability to alter the size of the operand is a distinguishing feature over the prior art. Starting with the specification:

> The microprocessor instruction decoder figures out the width of the operand field by the location of the instruction op-code in the four bytes. The compiler or assembler will normally use the shortest operand required to reach the desired address so that the leading bytes can be used to hold other instructions.

21:18-23. Under the "Variable Width Operands" heading the patentees noted:

> The advantage of this technique is the saving of a number of op-codes required to specify the different operand sizes in other microprocessors.

16:24-26.

The prosecution history is consistent with this teaching in the specification. The patentees highlighted the need for variable width operands to distinguish over the prior art.

> This [prior art] contrasts with the system of invention, in which instructions as well as **operands of variable-length** are loaded into instruction register 108. See e.g., the specification at pages 42-43, which describes the inclusion of 8, 16 and 24-bit operands (YYYYYYYY).... The system of the invention further differs from [the prior art] in that, for instructions requiring operands, the instruction's position within the instruction register is indicative of the width of the associated operand. This feature of the invention is described a page 43, lines 13-15 [citing above referenced specification excerpt]

Ex. CC at 11-12 [4/8/96 Amendment].

In the second amendment, the patentees reprised the importance of variable width operands to distinguish over the prior art.

> In the instant case, a single op-code is used and the width of a variable length operand depends on where the instruction begins in the current instruction group.

Ex. DD at 12-13 [6/17/97 Amendment].

Plaintiffs' examples of "Load-Short-Literal" and "Fetch-Via-PC" are not relevant to the variable width discussion. The "Load-Short-Literal" pushes "the 8-bit value found in byte 4 of the current 4-byte instruction group onto the Parameter Stack." 29:22-24. Thus, this operation is already optimized by using the minimum number (1) of bytes (*i.e.*, byte 4) to represent the data. The "Fetch-Via-PC" similarly references a pre-defined 32-bits of data which is the largest amount of data available and cannot therefore be optimized.

C. <u>"Instruction Register."</u>

As discussed *supra* at 12-13, Defendants adopt and accept Plaintiffs' refinement to the definition of an "Instruction Register" which requires that it hold "instruction groups" that are different from "traditional single instructions:"

> Plaintiffs' construction also incorporates the important feature of the '584 patent that the <u>instruction register holds an "instruction group" rather than the traditional single instruction</u>. This construction is based on the language of claim 29 itself, which recited "providing *instruction groups* to said instruction register," implying that the IR receives and <u>holds a group rather than a single instruction as is conventional</u>.

Plaintiffs' Br. at 39 (underlined emphasis added).

Plaintiffs state (at 39) that "the instruction registers '[h]olds 4-byte instruction groups while they are being decoded <u>and executed</u>.'" 19:56-59. To simplify this issue, Defendants offer a modified version of Plaintiffs' proposal based on the intrinsic language cited by Plaintiffs:

Plaintiffs' proposal	Defendants' modified proposal
"the register that temporarily stores the instruction group whose instructions are currently being decoded by the control unit of the computer"	"the register that temporarily stores the instruction group <u>while it is</u> being decoded <u>and executed</u> by the computer"

Defendants' modified proposal merely reflects the specification passage cited by Plaintiffs.[7]

[7] To the extent Plaintiffs refuse this suggestion, it is difficult to understand how Plaintiffs incorporate the "decode" from the specification but not the rest of the same sentence. Moreover, Plaintiffs' construction requires knowledge

D. "said instruction groups include at least one instruction that, when executed, causes an access to an operand or instruction or both."

Defendants generally agree with Plaintiffs' construction except for an unwarranted additional limitation imposed by Plaintiffs:

Plaintiffs' Proposal	Defendants' Proposal
The instruction being executed causes the CPU to use an immediate operand or execute a second instruction which is not the next sequential instruction.	The instruction being executed causes the CPU to use data or execute a second instruction.

Plaintiffs provide no intrinsic support for this additional limitation and rely solely on the extrinsic declaration of Plaintiffs' expert, Dr. Despain (at 42), who discusses the "Skip" instruction. Plaintiffs' position – and its supporting extrinsic evidence – contradicts the specification which notes:

> *The SKIP instruction can be located in any of the four byte positions 420 in the 32-bit instruction register 108*…. SKIP will jump over the remaining one, two, or three 8-bit instructions in the instruction register…

14:19-22. Notably, Dr. Despain's declaration discusses the underlined portion above but crops off the *italicized* portion. Placing the SKIP instruction in the 4th byte of the "four byte positions" would necessarily cause the processor to "execute a second instruction" which is the next sequential instruction (*i.e.*, the first instruction of the subsequent set of instructions). Additionally, the specification states "SKIPS may also be used in situations when no use can be made of the remaining bytes in a 4-instruction group." 23:12-14. This type of SKIP also causes access to the "second instruction" which is the next sequential instruction.

Plaintiffs should not be permitted to add a limitation by relying solely on an expert declaration that crops and ignores analysis of intrinsic evidence contrary to the declaration and eliminates embodiments described in the specification.

---

of the meaning of "decoded by the control unit" which begs the further questions of what is "decoded" and what is a "control unit." If Plaintiffs refuse this modification, it is more helpful to a jury to explain that the instruction register "contains or is connected to circuits that interpret the instructions" as proposed by Defendants.

E. "said operand or instruction being located at a predetermined position from a boundary of said instruction groups" and "decoding said at least one instruction to determine said predetermined position."

The parties have two disputes with respect to these phrases: (1) which instruction group has the "predetermined position" – the "accessing" group or the "accessed" group; and (2) Plaintiffs' unwarranted negative limitation that the "predetermined position" is "determined … without reference to operand or address bits in the accessing instruction."

With respect to "which" group has the predetermined position, Plaintiffs incorrectly state that the specification provides no guidance. The Abstract explains what this phrase means:

> A high-performance microprocessor system using instruction that access operands and instructions located relative to the current instruction group rather than located relative to the current instructions, as is the convention, is disclosed herein.

"Current" refers consistently to the "instruction that access[es]." The "current instruction group" is the instruction group containing the instruction that is currently executing. Defendants' construction reflects an ordinary usage of "current." Interestingly, Plaintiffs themselves use the term "current group" for the Microloop instruction to be the group currently in the instruction register being executed (at 44, line 6).

Plaintiffs cite no express intrinsic record definition to support their position but instead rely upon a theory of definition-by-consistent-operation. According to Plaintiffs, the fact that "seven instructions" all consistently operated in a certain fashion is enough support to define a term with that operation. If such a construction methodology is appropriate for this term, it should similarly be applied elsewhere including the "groupedness" discussion *supra* at 8-11. It should not be selectively applied to only those definitions proffered by Plaintiffs.

Plaintiffs' negative limitation of "without reference to operand or address bits in the accessing instruction" is more troubling. Plaintiffs cite no intrinsic evidence for this negative limitation. Notably, during the extensive discourse with the PTO, the patentees never cited "without reference to operand or address bits in the accessing instruction."[8]

[8] This is yet another example of where Plaintiffs take inconsistent positions regarding the use of negatively-phrased definitions. When it suits their case, Plaintiffs argue that "negatively-phrased definitions are unhelpful to one of

Moreover, Plaintiffs' position regarding "predetermined" contradicts the words of the claim and at least one of the express definitions given those words in the intrinsic record. Claim 29 requires "decoding said at least one *instruction* to determine said position." In the prosecution history, the patentees stated that "the term *instruction* is used to mean both the entire encoding including the immediate operand or a sub-part of the whole excluding the immediate operand." Ex. FF at 9 [11/26/97 Amendment]. Thus, decoding the "instruction" includes two options: (1) decoding the entire encoding including the immediate operand; or (2) a sub-part excluding the immediate operand. Plaintiffs' construction would exclude option (1) in direct contradiction to the intrinsic record without any basis.

F. "locating said predetermined position."

The parties have three primary disputes with respect to this term: (1) Plaintiffs' definition makes the definition of the asserted claim circular and redundant; (2) a determination of which instruction group is referenced; and (3) the specification clearly states that effective addresses are calculated at "assembly or linking time" and not "at run time." Issue (2) is addressed *supra* in section II.E and is not repeated in this section.

First, Plaintiffs' definition makes the last two claim terms circular:

Claim Language	Plaintiffs' Construction
locating said predetermined position	Establishing operand or instruction **supply**…
supplying, from said instruction groups, using the predetermined location, said operand or instruction or both to said central processing unit	(Undisputed by parties): using the results of the **locating** step…

Under Plaintiffs' definition, the "locating" clause would use "supply" to define itself and the "supply" clause would use "the results of the locating step." Such circularity is distinctly unhelpful to a jury. Moreover, it is unclear what "establishing … supply" means.

---

ordinary skill in the art attempting to determining the (positive) scope of the claims," Pl. Br. at n.9, even when such negative limitations are compelled by the intrinsic record. Here, however, Plaintiffs seek to add a negative limitation without citing *any* support from the intrinsic record.

The intrinsic record does not support Plaintiffs' construction. The "locating" step was added in the final amendment with little comment. The specification contains no guidance on the meaning of "locating" or "location."

Claim 29 provides some guidance on "locating" and/or "location." The claim references a "predetermined position," which both parties seem to agree means a position relative to some instruction group (*i.e.*, the beginning or end) and which is "determined" by the "decoding" step. Since "locating" and "location" are distinct terms from "predetermined position," the definition of "locating" and "location" is presumptively different. One explanation is that "locating" and "location" reference the address of the "predetermined position." Defendants' proposal reflects this explanation by defining the term to mean "[u]sing the results of the decoding step to ascertain the address of the addressed operand or instruction…"

In ascertaining this address, the specification distinguishes the invention over the prior art:

> In many computers, the effective address is calculated by adding or subtracting an operand with the current Program Counter. This math operation requires from four to seven machine cycles to perform and can definitely bog down machine execution. The microprocessor's strategy is to perform the required math operation at assembly or linking time and do a much simpler "Increment to next page" or Decrement to previous page" operation at run time.

20:43-50. The point of Defendants' proposed construction is that the '584 specification requires that this "math operation" not be done at run time, but rather during the assembly/linking phase.[9] Plaintiffs oppose this unambiguous language with a specification passage stating that "the microprocessor 50 next address logic treats the three operands similarly by adding or subtracting them to the current program counter" (11:13-16). When understood in the context of claim 29, the two statements are consistent, do not conflict, and support Defendants' proposal.

In plain English, Plaintiffs' citation to 11:13-16 explains that additions and subtractions are performed. Defendants' citation to 20:43-50 explains "when" those additions and

[9] "Run time" is during actual execution (*i.e.*, "running") of the instructions. "Assembly or linking time" is during the process of building the program into executable instructions.

subtractions are done – only at assembly/linking (not run time). Claim 29 covers actions occurring at run time – actions which do not include these additions and subtractions. Therefore, Defendants' construction is more consistent with claim 29 as interpreted by the intrinsic record.

G. "Microprocessor system."

Only the start of the preamble in claim 29 references "microprocessor system" to give a descriptive context / use and the body of the claim never references the term. A preamble is not limiting "where a patentee defines a structurally complete invention in the claim body and uses the preamble only to state a purpose or intended use for the invention." *Rowe v. Dror*, 112 F.3d 473, 478 (Fed. Cir. 1997); *IMS Tech., v. Haas Automation, Inc.*, 206 F.3d 1422, 1434 (Fed. Cir. 2000)("The phrase 'control apparatus' in the preamble merely gives a descriptive name to the set of limitations in the body of the claim that completely set forth the invention"). The preamble term "microprocessor" is not limiting and need not be construed by the Court for the '584 patent.

Plaintiffs' reference to the '336 patent construction is misguided because the *bodies* of asserted claims of the '336 patent actually recite "microprocessor" (unlike asserted claim 29 of the '584 patent). However, to the extent that "microprocessor system" is limiting, it has the same meaning as in the '336 patent.

III. CONCLUSION

For the foregoing reasons, Defendants request that the Court adopt their proposed claim constructions as expressed in Exhibit A.

Submitted on behalf of all Defendants

__/s/_____

David J. Lender, Esq.
Matthew Antonelli, Esq.
Tarra L. Zynda, Esq.
Weil Gotshal & Manges LLP
767 Fifth Avenue
New York, NY 10153-0001
david.lender@weil.com
matthew.antonelli@weil.com
tarra.zynda@weil.com

David J. Healey, Esq.
Weil Gotshal & Manges LLP
700 Louisiana Street, Suite 1600
Houston, TX 77002
david.healey@weil.com

Eric H. Findlay, Esq.
Ramey & Flock PC
100 E. Ferguson, Suite 500
Tyler, TX 75702
efindlay@rameyflock.com

Harry Lee Gilliam, Jr., Esq.
Gilliam & Smith LLP
303 South Washington Avenue
Marshall, TX 75670
gil@gilliamsmithlaw.com

Counsel for Matsushita, JVC and Panasonic Defendants

Scott F. Partridge, Esq.
Baker Botts LLP
One Shell Plaza
910 Louisana Street, Suite 3000
Houston, TX 77002-4995
Scott.partridge@bakerbotts.com

Carl R. Roth
Texas Bar No. 901984225
cr@rothfirm.com
Michael C. Smith
Texas Bar No. 900641877
ms@rothfirm.com
THE ROTH LAW FIRM, P.C.
115 North Wellington, Suite 200
P.O. Box 876
Marshall, Texas 75671
Tel: (903) 935-1665
Fax: (903) 935-1797

Counsel for Toshiba Defendants

John J. Feldhaus, Esq.
Anthony Hyeok Son, Esq.
Matthew A. Smith, Esq.
Foley & Lardner
3000 K Street NW, Suite 500
Washington, DC 20007-5111
Nec-tpl@foley.com

Guy N. Harrison, Esq.
P.O. Box 2845
217 N. Center
Longview, TX 75605
Cj-gnharrison@att.net

Counsel for Defendant NEC Electronics

James H. Wallace, Jr.
DC Bar. No. 016113
jwallace@wileyrein.com
Gregory E. Lyons
DC Bar. No. 436071
glyons@wileyrein.com
Kevin P. Anderson
DC Bar. No. 476504
kanderson@wileyrein.com
WILEY REIN LLP
1776 K Street, N.W.
Washington, D.C. 20006
Tel: (202) 719-7000
Fax: (202)719-7049

Carl R. Roth
Texas Bar No. 901984225
cr@rothfirm.com
Michael C. Smith
Texas Bar No. 900641877
ms@rothfirm.com
THE ROTH LAW FIRM, P.C.
115 North Wellington, Suite 200
P.O. Box 876
Marshall, Texas 75671
Tel: (903) 935-1665
Fax: (903) 935-1797

Counsel for ARM Defendants